Block 5: Miscellaneous

1. Split-validation, k-fold cross-validation and Bootstrap sampling.
2. Bayesian classifiers.
3. Performance measures for classification.
4. Decision making under uncertainty: Decision Trees.

1. Split-validation, K-fold cross-validation and Bootstrap sampling

The validation process provides information on the accuracy of the predictions.

Up to now we assume that the data set D with N cases has been randomly partitioned into:

a *training set* T, and a *validation set* V,

such that they are disjoint using, for instance, the rule 80-20%.

Thus, none of the validation cases are used for training and can be considered to be new to the BN model.

After training from T, we will apply BN to the cases from V in order to determine *accuracy of predictions*.

This procedure is know as SPLIT-VALIDATION.

- k-fold cross-validation:

Cross-validation is a way to predict the fit of a model to a validation set.

With k-fold cross-validation we reuse the data set D generating k splits of D into non-overlapping training and validate sets T_s, V_s, s=1,…,k, with approximately proportions of data (k-1)/k and 1/k, respectively.

For each split we obtain the accuracy of the predictions as with split-validation. From these k values we can, then, compute the mean, as well as the standard deviation.

Although k can be set to any number, by far, the most common convention is to use **10-fold cross-validation** The reason is that the empirical evidence suggests that there is little added benefit in using a greater number.

For each of the 10 folds (each comprising 10 percent of the total data), a Bayesian network model is built on the remaining 90 percent of data. The fold's matching 10 percent sample is then used for model evaluation. After the process of training and evaluating the model has occurred for 10 times (with 10 different training/testing combinations), the average performance across all the folds is reported.

Of particular interests is the case k=N, named **leave-one-out cross-validation**. In this case, the accuracy of the predictions has to be computed globally, taking into account the successes and failures, and not for any "fold" separately, since we only have one case to predict for each "fold".

- Bootstrap sampling:

A slightly less frequently used alternative to K-fold CV is known as **bootstrap sampling**. Generally speaking, these refer to the statistical methods of using random samples of data to estimate the properties of a larger set.

When this principle is applied to machine learning model performance, it implies the creation of several randomly selected training and test datasets, which are then used to estimate performance statistics. The results from the various random datasets are then averaged to obtain a final estimate of future performance.

So, **what makes this procedure different from K-fold CV?**

Whereas cross-validation divides the data into separate partitions in which each example (case) can appear only once, the bootstrap allows examples to be selected multiple times through a process of **sampling with replacement**.

This means that from the original dataset of N examples, the bootstrap procedure will create one or more new training datasets that will also contain N examples, some of which are repeated. The corresponding test datasets are then constructed from the set of examples that were not selected for the respective training datasets.

2. Bayesian Classifiers

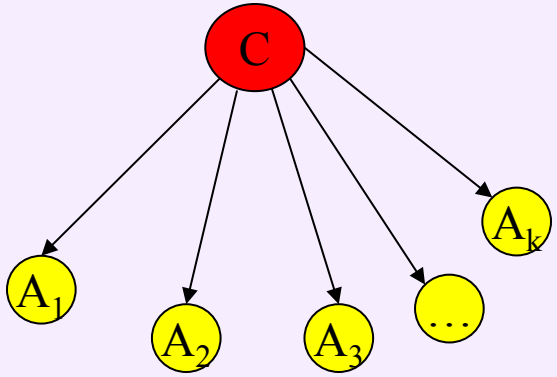
One of the main reasons for learning a Bayesian network is to use it as a "classifier". Classification is a very important task in data analysis and pattern recognition, and requires the construction of a classifier, which is a function that assigns a class to a new case, based on values that take a set of attributes or features in this case.

Obtaining a classifier from a set of pre-classified data (cases) is one of the key problems of machine learning. Different approaches:

- K-Nearest Neighbours,
- Neuronal Networks,
- Decision trees,
- Random Forest,
- Support Vector Machine,
- ...
- Bayesian Network Classifiers: Naive Bayes, Augmented Naive Bayes, Tree-Augmented Naive Bayes (TAN), Unrestricted Bayesian Network, Selective Naive Bayesian, ... (they are all white-box models!).

① Naive Bayes classifier

It is one of the most effective classifiers (in the sense of predictive behavior). Learn from the set of training data T the probability distribution of each Ai attribute, given class C. The structure is fixed ahead in this classifier, and is:

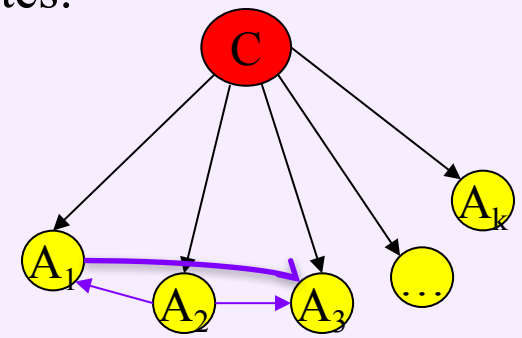


The classification of a new case, based on the attributes A_1, ..., A_k, is done by applying Bayes' Theorem to calculate the probability of C conditioned on the attributes, and by classifying the case with that classe that maximizes the (a posterior) probability.

Markov's condition implies assuming that all attributes are conditionally independent among them given the value of class C. Although this assumption is strong and unrealistic, the classifier behaves surprisingly well!

② Augmented Naive Bayes

This is a classifier based on the Naive Bayes structure. The variable class C must be, as in the other case, parent of all the attributes, but now, in addition, additional arcs are allowed between the attributes:



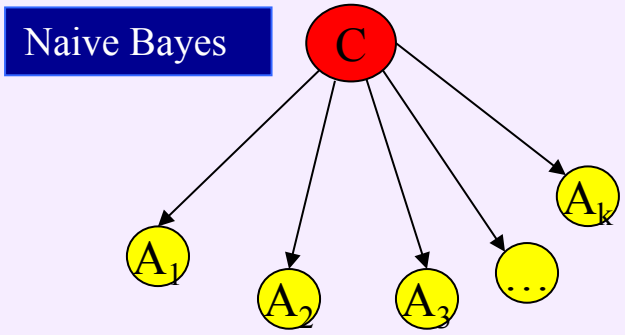
That is, this classifier models the interactions between the attributes through this structure over-imposed to the Naive Bayes structure.

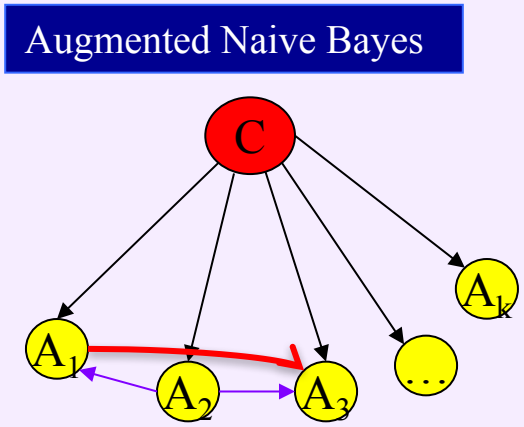
This extension implies additional computational costs. Unlike Naive Bayesi, in this case the structure of the BN must be learned (based on the data) but with some restrictions:

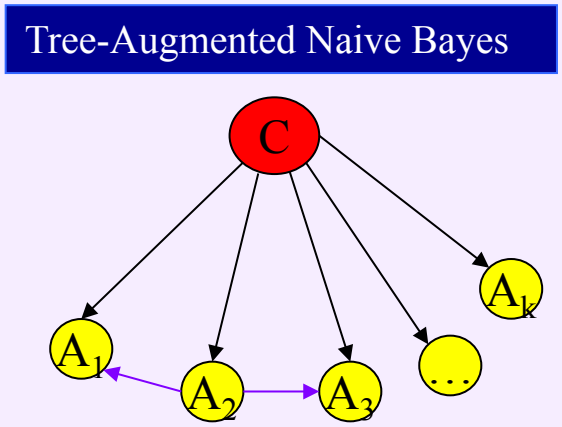
- Black list: there can be no directed arcs from attributes to the class C.
- White list: dierected arcs from C to each one of the attributes (like Naive Bayesian) are forced.

③ Tree-Augmented Naive Bayes (TAN)

This classifier is a particular case of the Augmented Naive Bayesian network in which each attribute has as parents class C and, at most, another attribute.







The Construct-TAN procedure (Friedman et al. 1997)

It is a procedure that follows the ideas of Chow and Liu (1996).

The Chou and Liu procedure raises the problem of constructing, from a dataset T, a **tree** (a **"tree"** is a DAG in which all variables have exactly one parent except the root variable, which does not have any), so that the likelihood function is maximized.

This is done by reducing the problem to find a *maximal weighted spanning tree* in a graph, consisting in the selection of arcs on the graph with n variables so that they form a tree and the sum of the associated weights of these arcs is maximum. There are known algorithms that work very well to do this task.

The **Construct-TAN procedure** adjusts the Chou and Liu procedure to find not a single tree but the TAN structure that maximizes the likelihood function.

The Construct-TAN procedure uses the Conditional Mutual Information between attribute pairs given the variable class C: MI (A_i, A_j / C), which measures the information that A_j contributes about A_i (or vice versa) when it is known value of C.

$$MI(A_i,\ A_j/C) = E_C(I(A_i,\ A_j)) =$$

$$\sum_c P(C=c)\Big(\sum_a\sum_b P(A_i=a, A_j=b/C=c)\ \log \frac{P(A_i=a, A_j=b/C=c)}{P(A_i=a/C=c)\,P(A_j=b/C=c)}\Big)$$

$$= \sum_a\sum_b\sum_c P(A_i=a, A_j=b, C=c)\ \log \frac{P(A_i=a, A_j=b, C=c)\,P(C=c)}{P(A_i=a, C=c)\,P(A_j=a, C=c)}$$

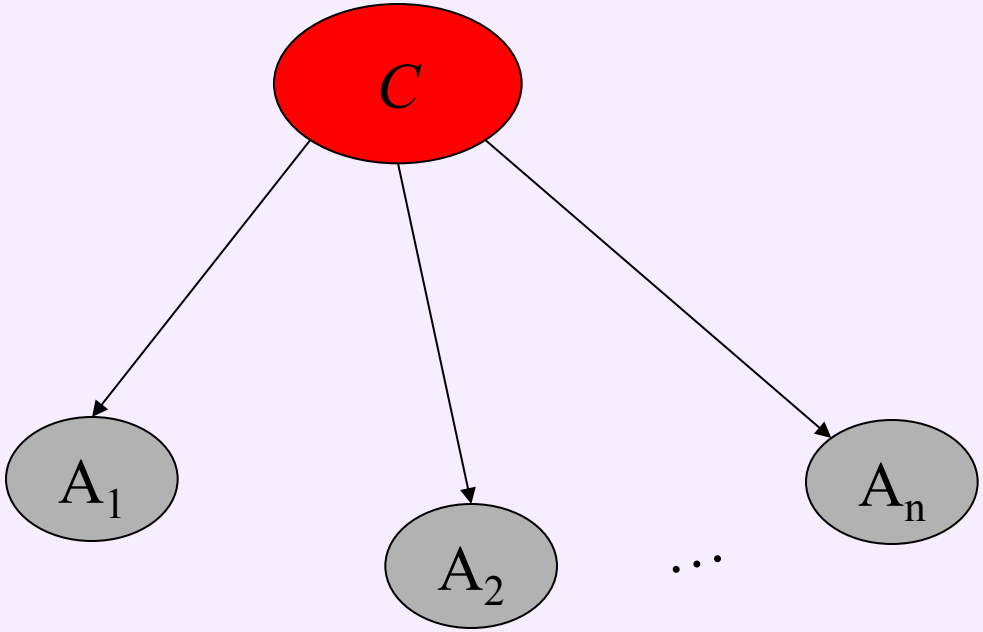
The steps are:

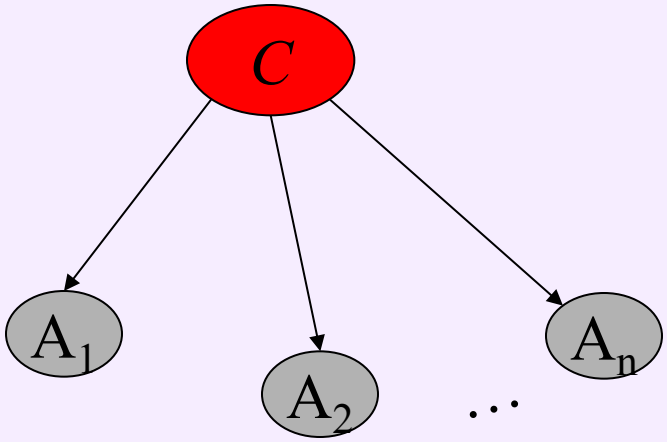
1. Calculate MI (Ai, Aj / C) for every, i, j = 1, ..., n, and i ≠ j.
2. Build a non-directed graph in which the nodes are variables A_1, ..., A_n, giving to the connection between nodes A_i and A_j (unoriented arc) a weight equal to MI (Ai, Aj / C).
3. Build the *maximal weighted spanning tree* with a known algorithm.
4. Transform the resulting undirected tree into a directed one by choosing a node as root and directing the arcs "outwards" from it.
5. Build a TAN model by adding the root node C and a directed arc from C to each Ai.

THEOREM: (Th. 2, Friedman, Geiger & Goldsamidt, "Bayesian Network Classifiers", Machine Learning 29, 131-163, 1997).
The Construct-TAN procedure results in an optimal TAN that maximizes the likelihood function of all possible TANs.

3. Performance measures for classification

Assume we have learned a Bayesian Classifier (a BN of a particular type: **diagnosis**) in which we have some features $A_1,\ldots,A_n$, and a class variable C, from the training data set T (split-validation).

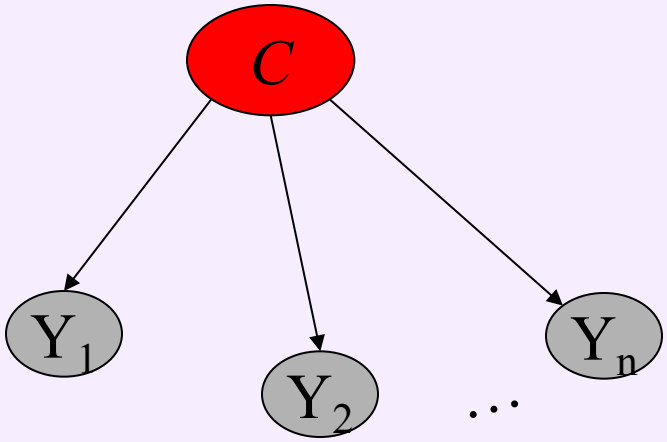




Now, for each case in the validate data set V, from the values (evidences) of variables $A_1,\ldots, A_n$, we have to assign to the case a predefined categorical class, say $C_1,\ldots,C_r$.

Classification falls into one of the following tasks:

- Binary: the input is to be classified into one, and only one, of two non-overlapping classes. It is is the most popular classification task.
- Multi-class: the input is to be classified into one, and only one, of r non-overlapping classes.
- Multi-labelled: the input is to be classified into several of r non-overlapping classes.
- Hierarchical: the input is to be classified into one, and only one, class which are be divided into subclasses or grouped into super-classes.



Then, we obtain a "confusion matrix" in which in row we have the observed (recorded) class, and in column we have the predicted class.

Diagonal values correspond to correct classification.

Observed	Predicted C_1	$\cdots$	C_r
C_1	a_{11}	$\cdots$	a_{1r}
$\vdots$	$\vdots$	$\cdots$	$\vdots$
C_r	a_{r1}	$\cdots$	a_{rr}

M = number of cases in the validation set V.

Observed	Predicted		
	C_1	$\cdots$	C_r
C_1	a_{11}	$\cdots$	a_{1r}
$\vdots$	$\vdots$	$\cdots$	$\vdots$
C_r	a_{r1}	$\cdots$	a_{rr}

For each one of the classes C_j, j=1,…,r, we can construct from the confusion matrix a matrix like that:

Observed	Predicted	
	C_j	C_j^c
C_j	t_p	f_n
C_j^c	f_p	t_n

Where $C_j^c = \{C_1,\ldots, C_{j-1}, C_{j+1},\ldots, C_r\}$.

$$t_p = a_{jj}$$

$$f_n = \sum_{\ell \neq j} a_{j\ell}$$

$$f_p = \sum_{m \neq j} a_{mj}$$

$$t_n = \sum_{m \neq j,\, \ell \neq j} a_{m\ell}$$

True positive

False negative

False positive

True negative

Observed	Predicted	
	C_j	C_j^c
C_j	t_p	f_n
C_j^c	f_p	t_n

tp = true positive (correctly recognized class cases),

fn = false negative (cases that are not recognized as class examples),

fp = false positive (cases incorrectly assigned to the class),

tn = true negative (correctly recognized cases that do not belong to the class).

The correctness of a classification can be evaluated by computing these values.

$$t_p + f_n + f_p + t_n = M$$

Observed	Predicted C_j	Predicted C_j^c
C_j	t_p	f_n
C_j^c	f_p	t_n

Measures for binary classification (or multi-class classification, for each class independently):

Accuracy	$\frac{t_p+t_n}{M}$	Overall effectiveness of a classifier
Error rate =1-Accuracy	$\frac{f_p+f_n}{M}$	Classification error rate
Precision	$\frac{t_p}{t_p+f_p}$	Class agreement of the data labels with the positive labels given by the classifier
Recall (sensitivity)	$\frac{t_p}{t_p+f_n}$	Effectiveness of a classifier to identify positive labels
Specificity	$\frac{t_n}{f_p+t_n}$	How effectively a classifier identifies negative labels

Measures for binary classification (or multi-class classification, for each class independently):

Accuracy	$\frac{t_p+t_n}{M}$	Overall effectiveness of a classifier
Error rate =1-Accuracy	$\frac{f_p+f_n}{M}$	Classification error rate
Precision	$\frac{t_p}{t_p+f_p}$	Class agreement of the data labels with the positive labels given by the classifier
Recall (sensitivity)	$\frac{t_p}{t_p+f_n}$	Effectiveness of a classifier to identify positive labels
Specificity	$\frac{t_n}{f_p+t_n}$	How effectively a classifier identifies negative labels

The F score: A measure of classifier performance that combines precision and recall into a single number using the **harmonic mean**, a type of average that is used for rates of change. The harmonic mean is used rather than the common arithmetic mean since both precision and recall are expressed as proportions between zero and one, which can be interpreted as rates.

Accuracy	$\frac{t_p+t_n}{M}$	Overall effectiveness of a classifier
Error rate =1-Accuracy	$\frac{f_p+f_n}{M}$	Classification error rate
Precision	$\frac{t_p}{t_p+f_p}$	Class agreement of the data labels with the positive labels given by the classifier
Recall (sensitivity)	$\frac{t_p}{t_p+f_n}$	Effectiveness of a classifier to identify positive labels
Specificity	$\frac{t_n}{f_p+t_n}$	How effectively a classifier identifies negative labels

The F score:

$$\frac{1}{\frac{1}{2}\left(\frac{1}{\text{precision}}+\frac{1}{\text{recall}}\right)} = 2\frac{\frac{t_p}{t_p+f_p}\times\frac{t_p}{t_p+f_n}}{\frac{t_p}{t_p+f_p}+\frac{t_p}{t_p+f_n}}$$

$$\frac{2\,t_p}{2\,t_p+f_n+f_p}.$$

Accuracy	$\frac{t_p+t_n}{M}$	Overall effectiveness of a classifier
Error rate =1-Accuracy	$\frac{f_p+f_n}{M}$	Classification error rate
Precision	$\frac{t_p}{t_p+f_p}$	Class agreement of the data labels with the positive labels given by the classifier
Recall (sensitivity)	$\frac{t_p}{t_p+f_n}$	Effectiveness of a classifier to identify positive labels
Specificity	$\frac{t_n}{f_p+t_n}$	How effectively a classifier identifies negative labels

The F score: $\frac{2\,t_p}{2\,t_p + f_n + f_p}$ Since it describes the model performance in a single number, it provides a convenient way to compare several models side by side. However, this assumes that equal weight should be assigned to precision and recall, an assumption that is not always valid. It is possible to calculate F-scores using different weights for precision and recall, but choosing the weights could be tricky at the best and arbitrary at worst.

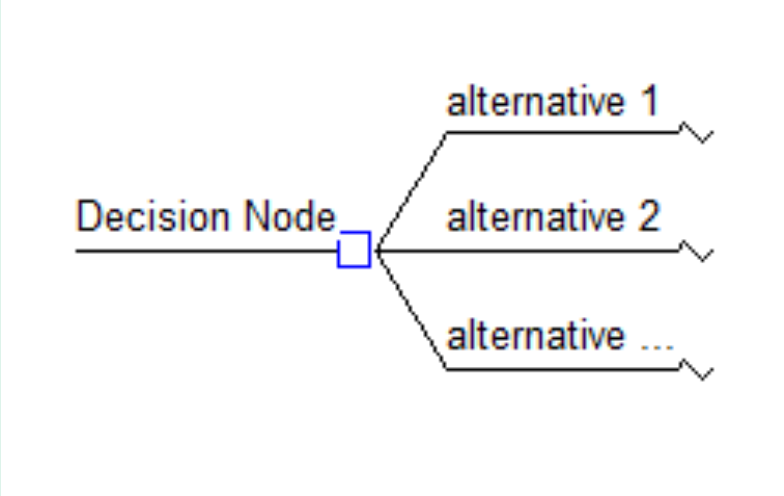
Measures for Measures for multi-class classification based on a generalization of the measures of binary classification for many classes:

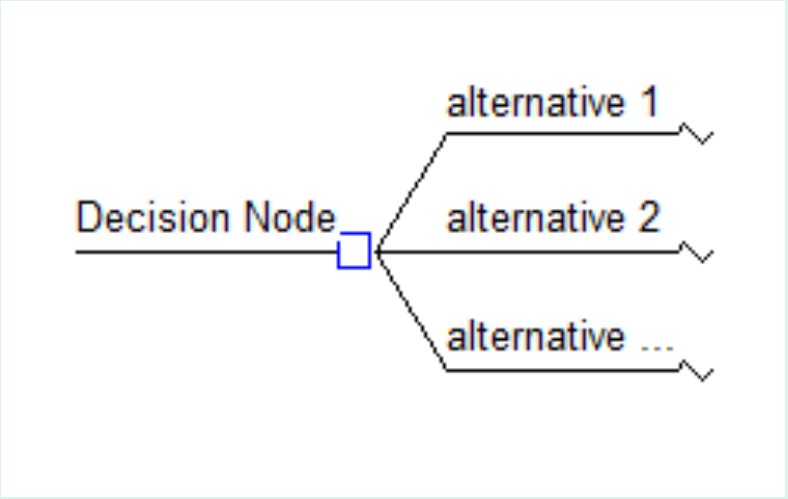
Average Accuracy	$\frac{\sum_{j=1}^{r} \frac{tp_j + tn_j}{M}}{r}$	The average per-class effectiveness of a classifier
Average Error rate =1-Average Accuracy	$\frac{\sum_{j=1}^{r} \frac{fp_j + fn_j}{M}}{r}$	The average per-class classification error
Average Precision	$\frac{\sum_{j=1}^{r} \frac{tp_j}{tp_j + fp_j}}{r}$	An average per-class agreement of the data class labels with those of the classifiers
Average Recall (sensitivity)	$\frac{\sum_{j=1}^{r} \frac{tp_j}{tp_j + fn_j}}{r}$	An average per-class effectiveness of a classifier to identify class labels
Average Specificity	$\frac{\sum_{j=1}^{r} \frac{tn_j}{fp_j + tn_j}}{r}$	An average per-class effectiveness of a classifier to identify negative class labels

4. Decision making under uncertainty: Decision Trees.

Decision trees are a modification of tree diagrams (a graph without cycles) so as to be able to represent a decision making process. The tree structure is in chronological order, generally left to right.

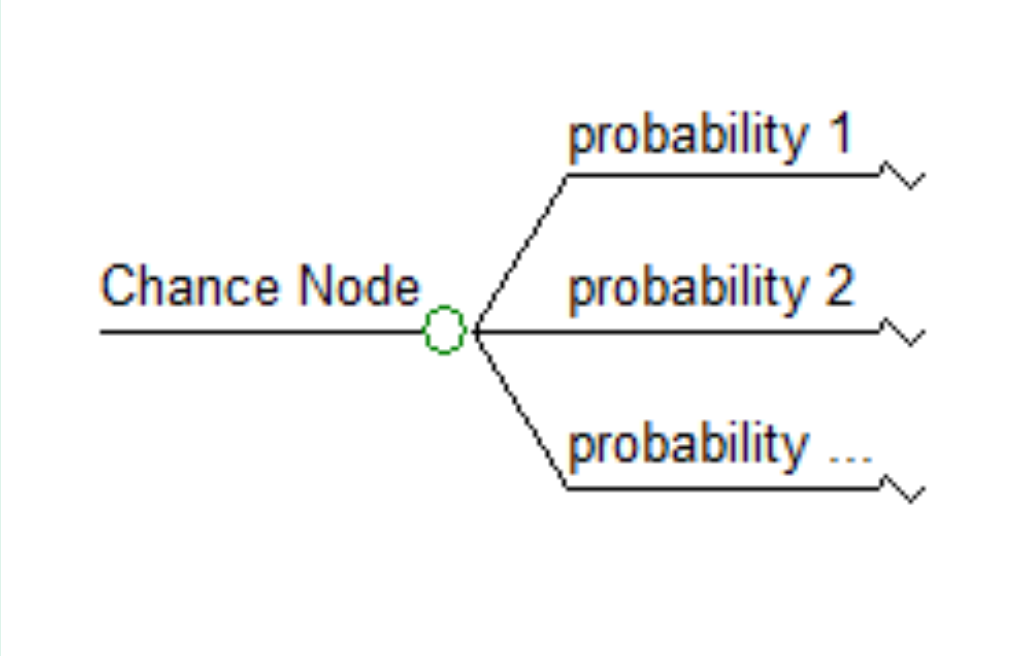
The decision tree uses two types of nodes, the decision node and the chance node. The decision node is represented with a square and denotes a decision the user must make.

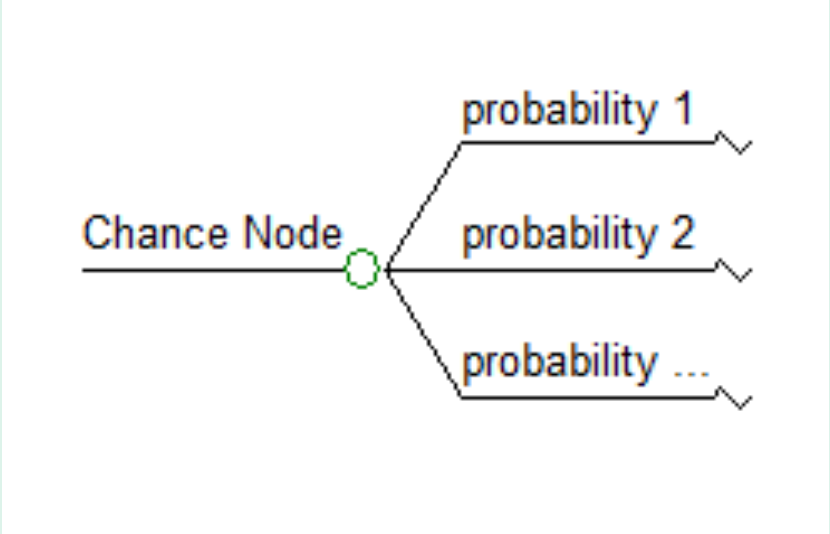




The branches from a decision node are the possible actions that can be taken from the decision, are called alternatives and must be mutually exclusive and exhaustive so that all possible outcomes are considered. The possible outcomes may be subjective and depend on the knowledge of the decision maker.

The second type of node is the chance node, which is represented with a circle. A chance node represents a random variable. It may also be referred to as a random node or uncertainty node.

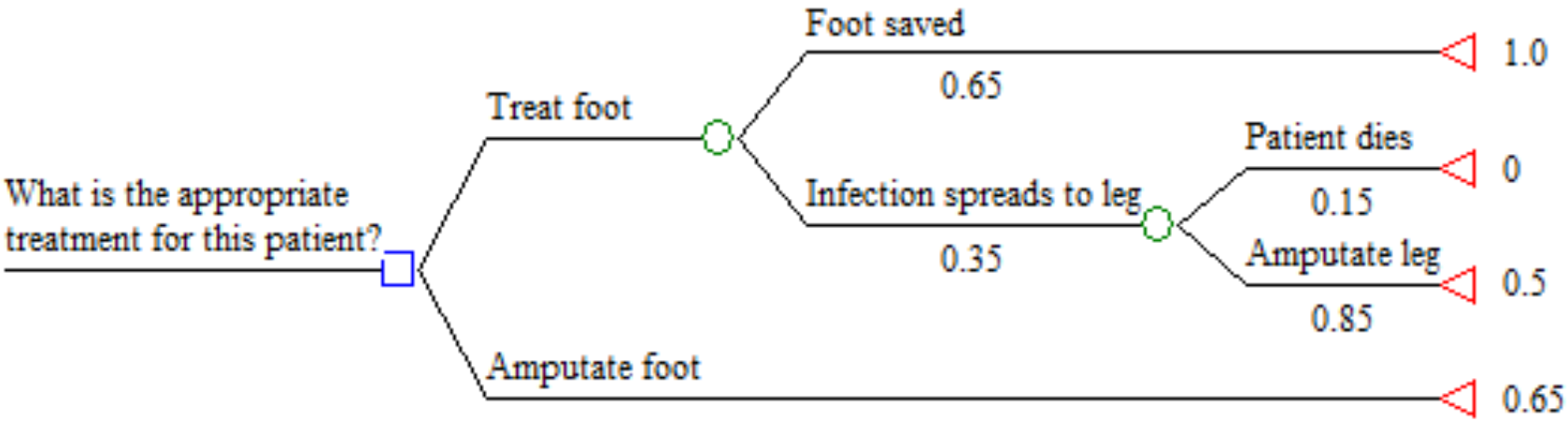




Each possible outcome the random variable can take must have a correspondence to one and only one edge produced. The sum of the probabilities of all the subsequent branches must equal one. In other words the definition of a random variable must hold for the node and its outcomes.

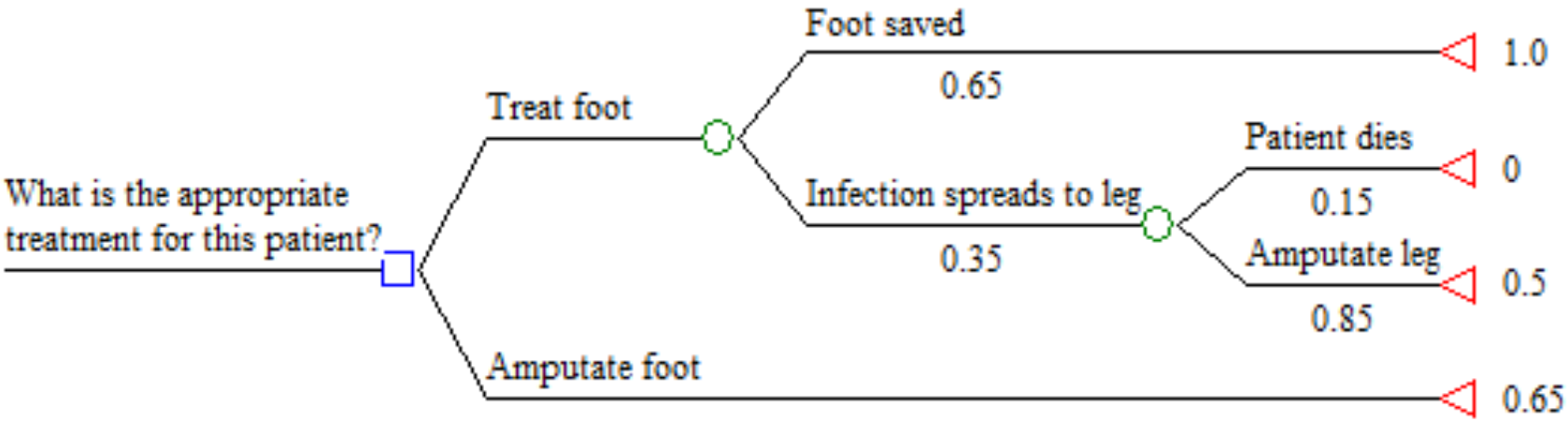
Each leaf (right most nodes with no children) of the tree represents the value of that particular outcome to the decision maker, which is called the **utility**.

Example: the injured rock climber.



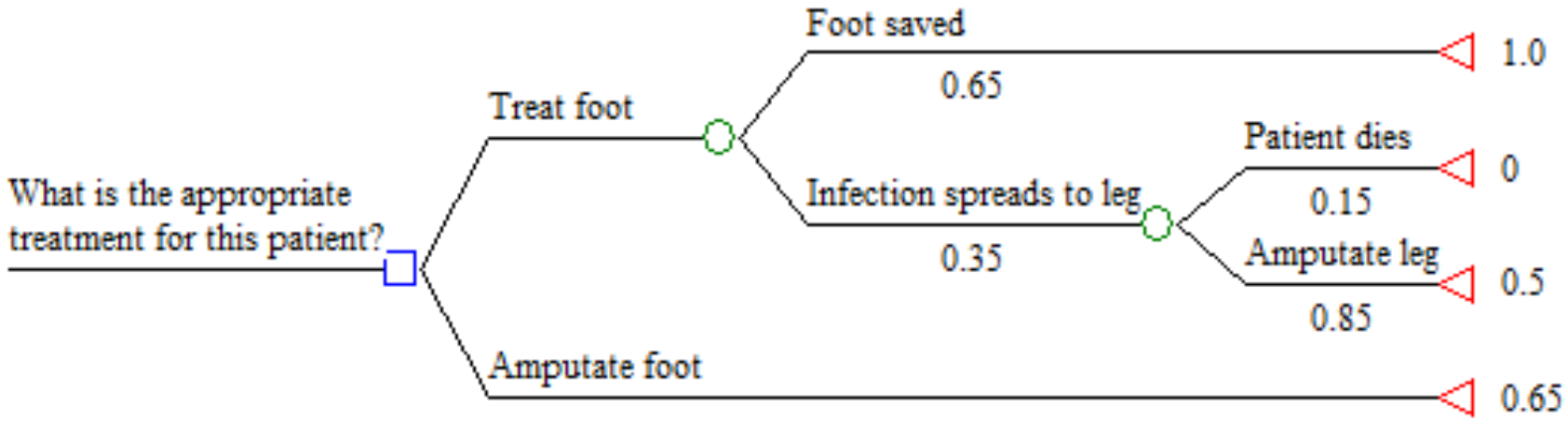
The root node is the decision of whether not to amputate the foot. Therefore, the alternatives are to treat the foot or to amputate the foot. The first chance node for the treatment alternative is whether the treatment worked thus saving the foot or if the infection spread.

Example: the injured rock climber.



There is a 65% chance it worked and a 35% it did not. The next chance node if the treatment failed is the patient dies (15%) or the leg is successfully amputated (85%). The utility values are at the leaves, and represented by red triangles.

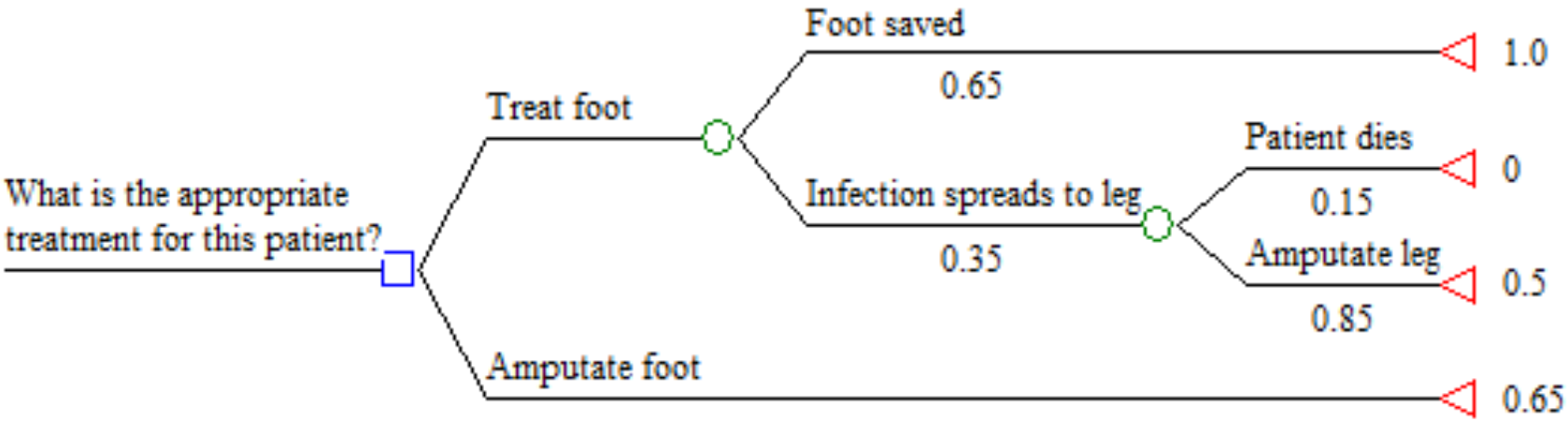
Example: the injured rock climber.



To summarize the probabilities:

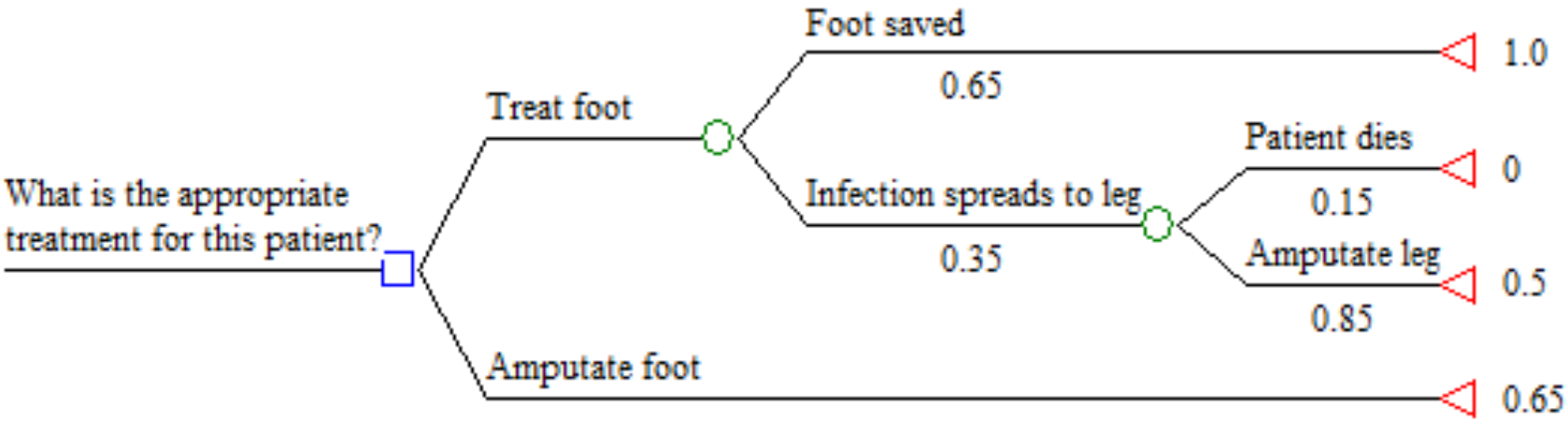
- $P(\text{foot saved}|\text{foot is treated}) = 0.65$
- $P(\text{infection spreads to leg}|\text{foot is treated}) = 0.35$
- $P(\text{patient dies}|\text{infection spreads to leg}) = 0.15$
- $P(\text{successful leg amputation}|\text{infection spreads to leg}) = 0.85$

Example: the injured rock climber.



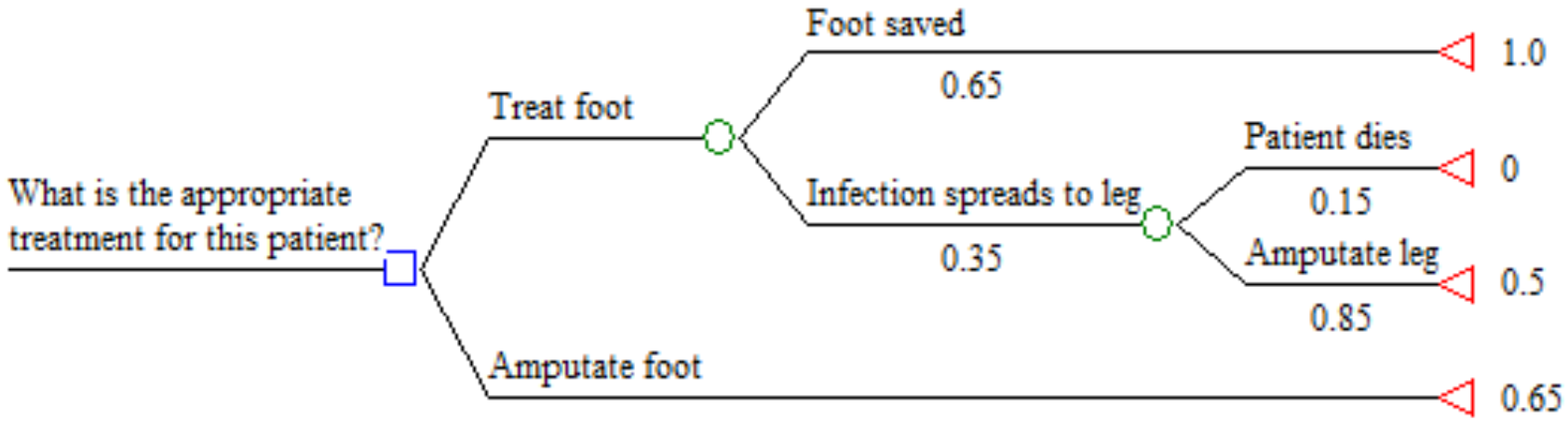
The **utility** is the subjective value assigned to each outcome at the leaves of the decision tree by the decision maker. It is the result of the specific action-outcome combination followed in the tree. The utility represents an amount of something defined by the processes outcome, examples are time and monetary units.

Example: the injured rock climber.



The expected utility of a chance node is the expected value the utility will take when all outcomes are considered. It is calculated by multiplying the utility times its probability of occurring from the particular path and summing over all the possibilities. The expected utility of a decision alternative is the expected utility of the subsequent node.

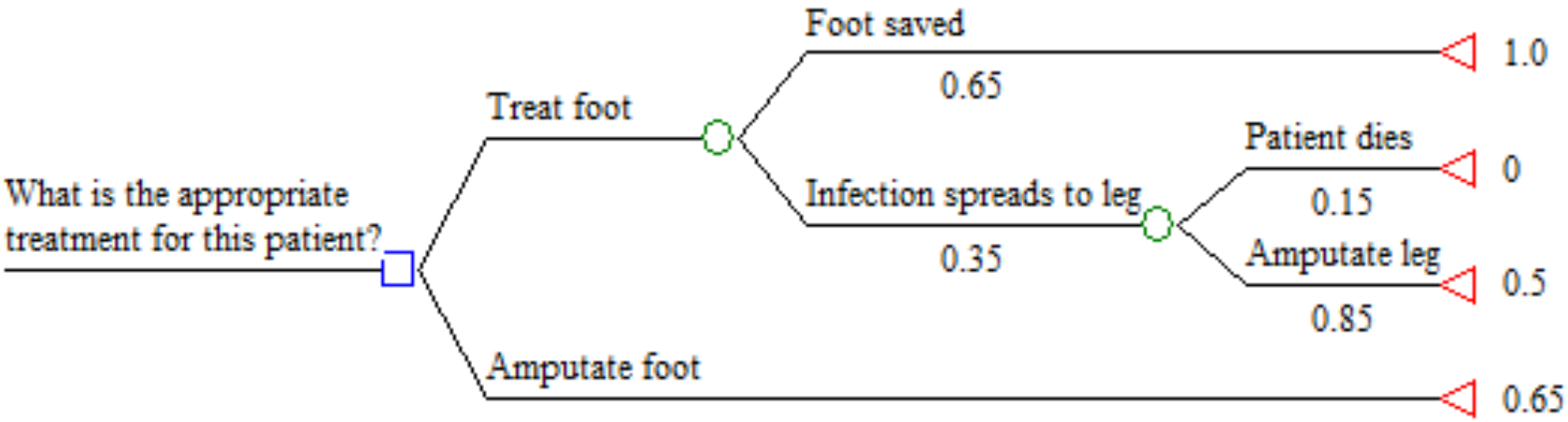
Example: the injured rock climber.



The utility values of the rock climber example appear to make sense; the patient dies is obviously the worst case therefore the utility of zero and the foot saved is the best case so the utility is one. In summary:

- $U(\text{foot saved}) = 1$
- $U(\text{amputate foot}) = 0.65$
- $U(\text{amputate leg}) = 0.5$
- $U(\text{patient dies}) = 0$

Example: the injured rock climber.

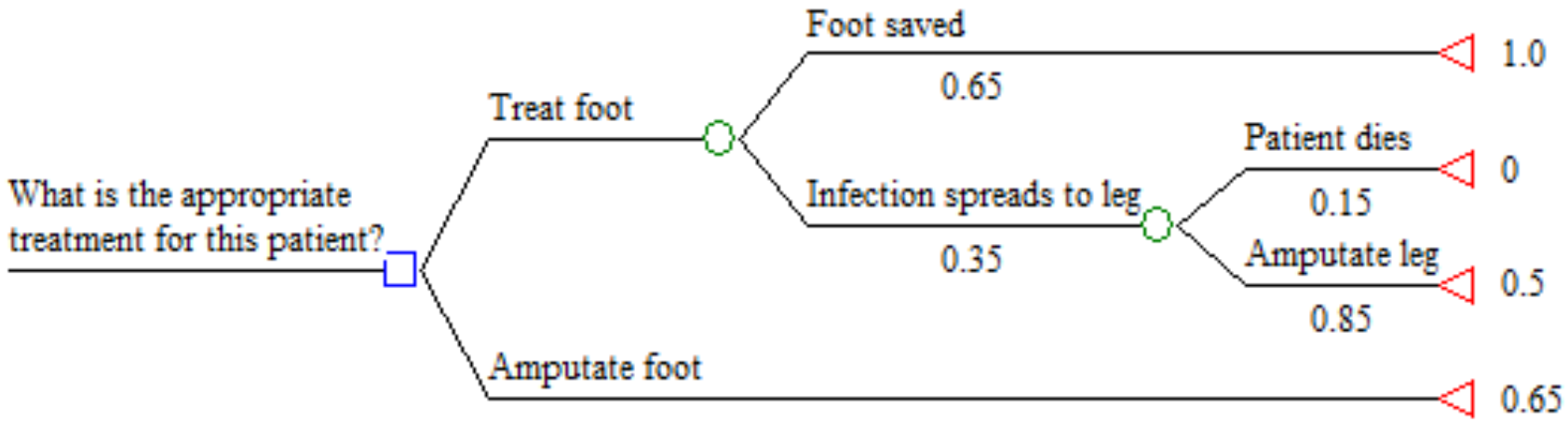


Calculation of the expected utility of the chance node for the spread of infection is:

$$EU\ (\text{infection spreads}) = 0.15(0) + 0.85(0.5) = 0.425$$

Now the value of the expected utility of the treat foot chance node can be found using the expected utility of the ancestor chance node in place of the utility:

Example: the injured rock climber.



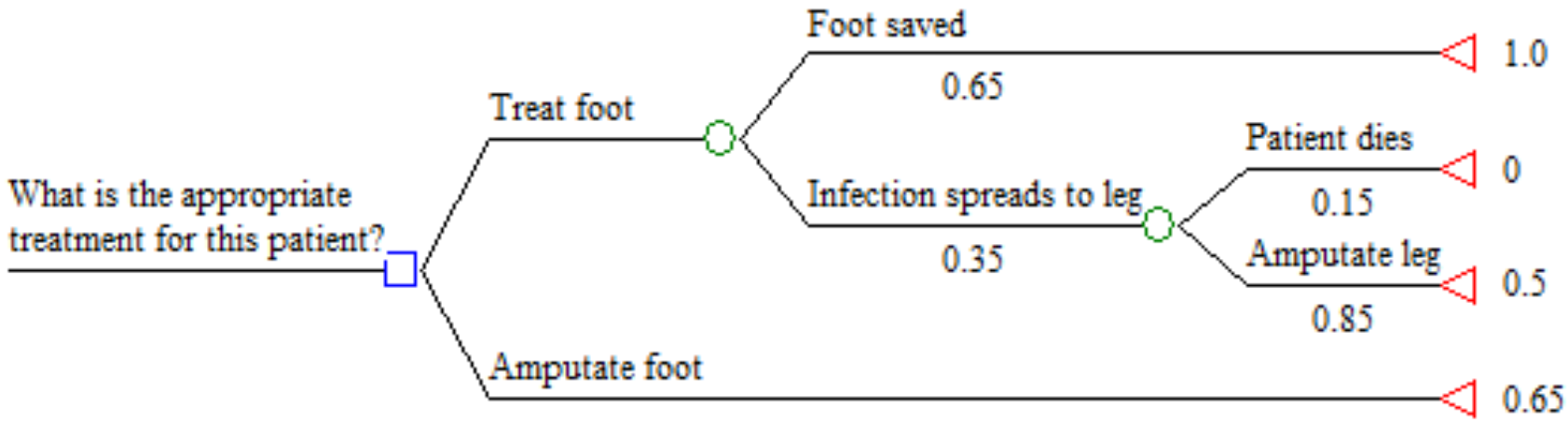
$$EU\,(\text{infection spreads}) = 0.15(0) + 0.85(0.5) = 0.425$$

$$EU\,(\text{treat foot}) = 0.65(1) + 0.35(0.425) = 0.79875$$

Since, the expected utility of the decision alternatives is the expected utility of the subsequent chance node:

$$EU\,(\text{Choose to treat foot}) = EU(\text{treat foot}) = 0.79875$$

Example: the injured rock climber.



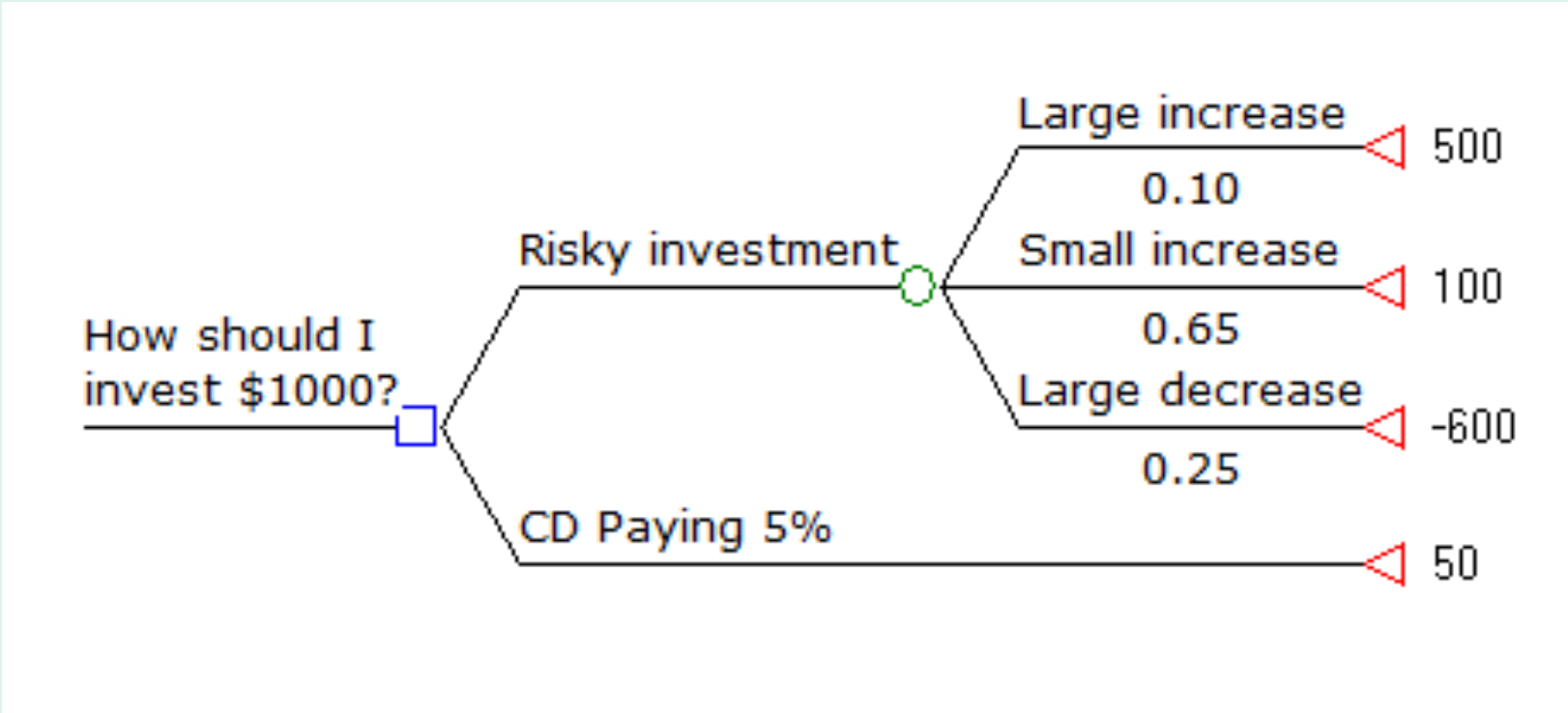
The expected utility of the root decision node can now be taken from the maximum of each alternative expected utility:

$$EU(\text{Root choice}) = max\,(EU(\text{treat foot}), EU(\text{amputate foot})) = max\,(0.79875, 0.65) = 0.79875$$

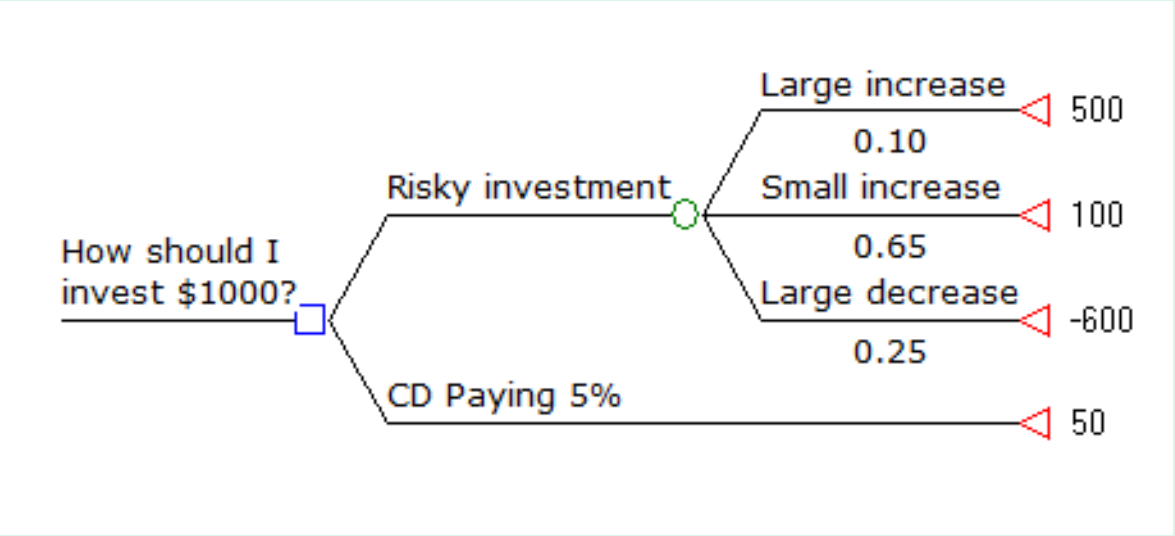
Therefore, the decision for the example would be to treat the foot.

Example: Stock Investment.

The decision tree's function is to choose whether it is better to invest in a CD (certificate of deposit) at 5%, or a stock that the decision maker has decided has the likely values shown.



Example: Stock Investment.



First it is necessary to look at the expected utility of the risky investment chance node.

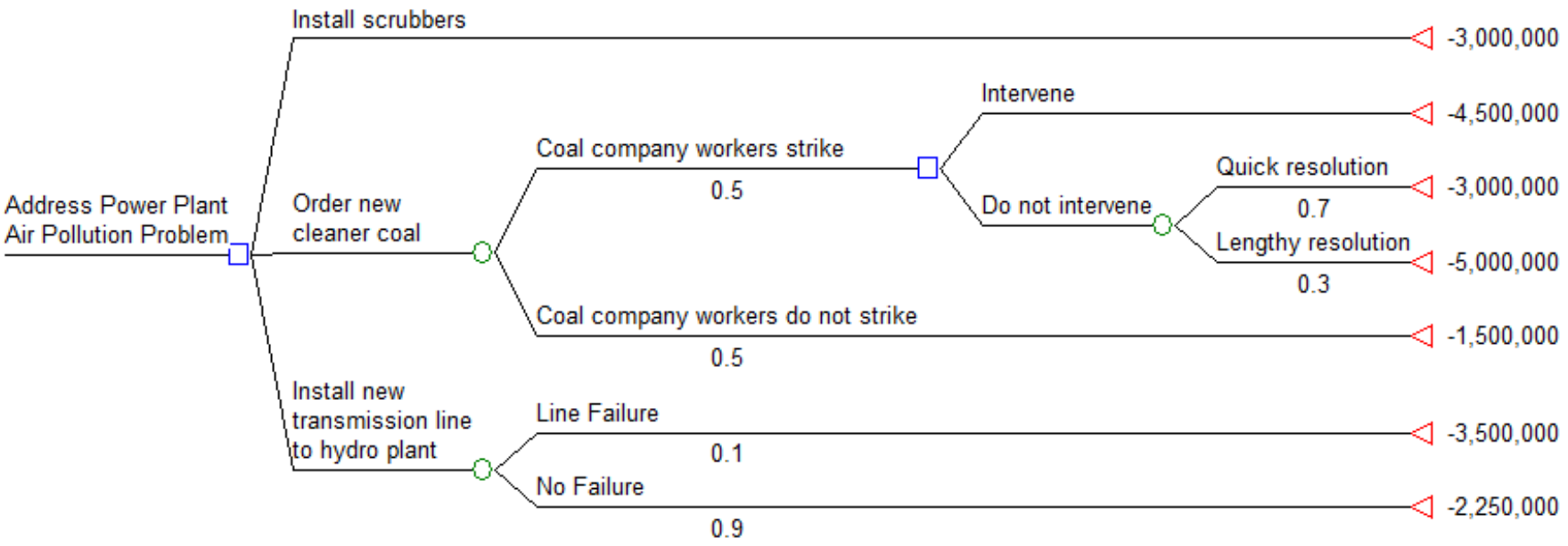
$$EU\,(\text{Risky investment}) = 0.10(500) + 0.65(100) + 0.25(-600) = -35$$

Now look at the expected utility of decision node considering the expected utility of each alternative.

$$EU\,(\text{How should I invest?}) = max\,(-35, 50) = 50$$

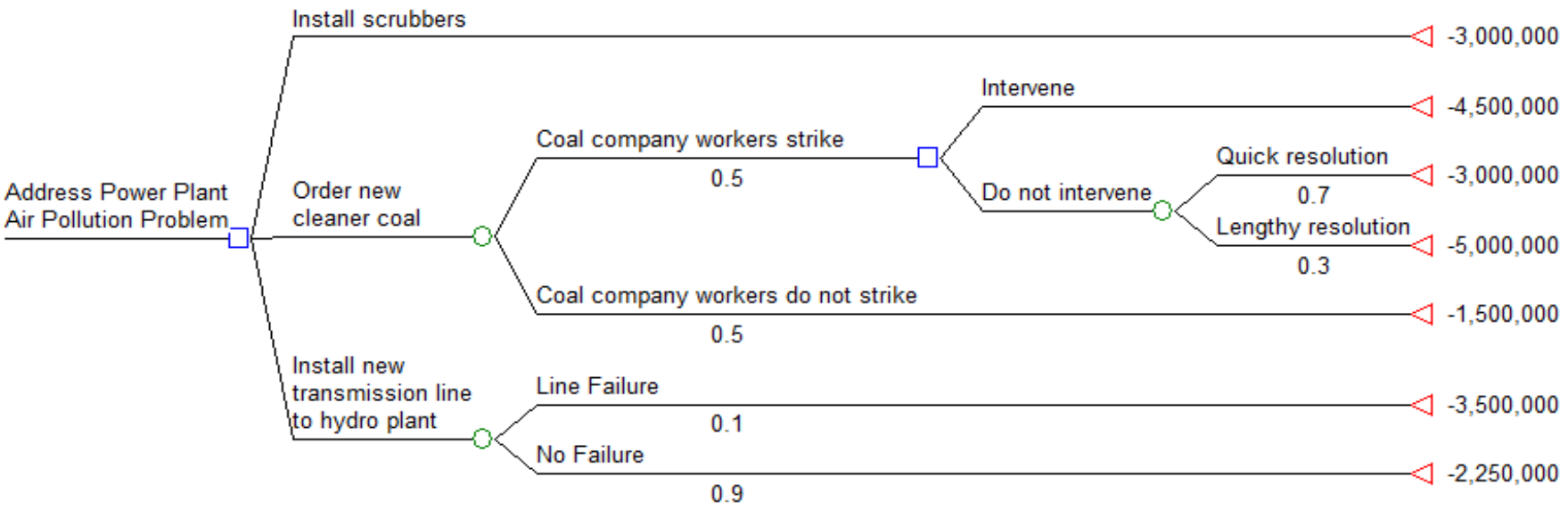
Therefore, the investment should be made into the CD.

Example: Power Plant Air Pollution Problem



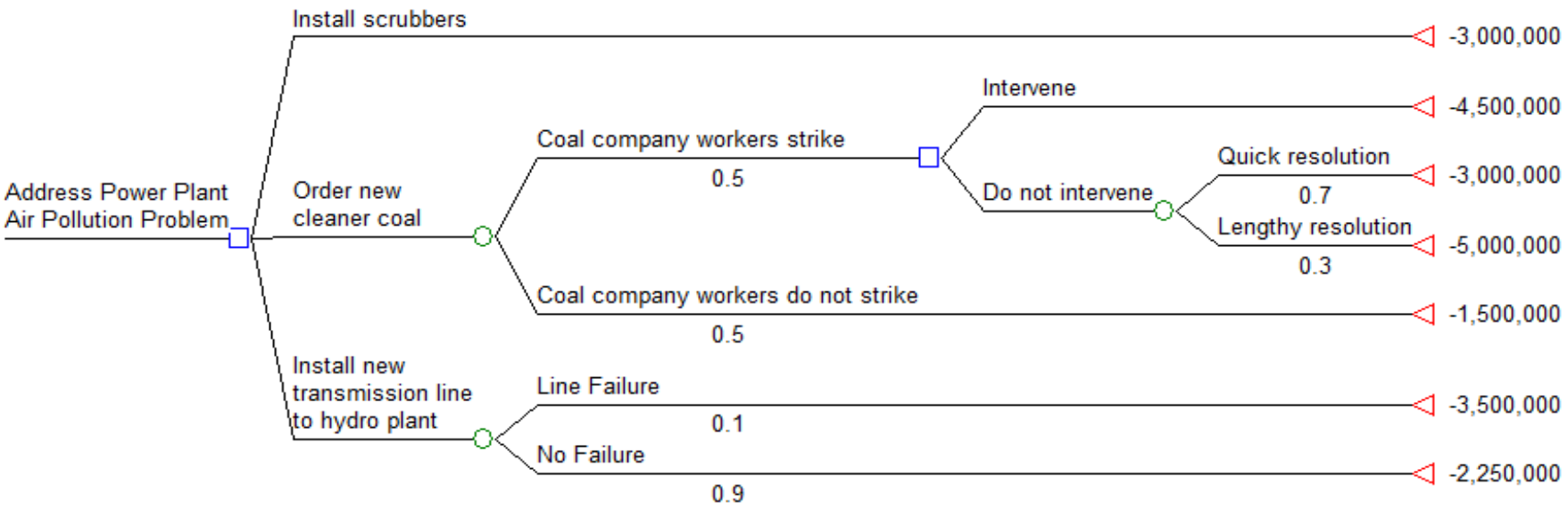
The problem is that a power plant is presented with new air pollution regulations it must meet. There are three options to consider installing: new scrubbers, ordering new cleaner coal and constructing a new transmission line to a hydro plant with excess capacity.

Example: Power Plant Air Pollution Problem



The company that would supply the new cleaner coal has a chance that the workers will go on strike. The utility company can decide whether or not to intervene resolving the situation but at an increased cost.

Example: Power Plant Air Pollution Problem

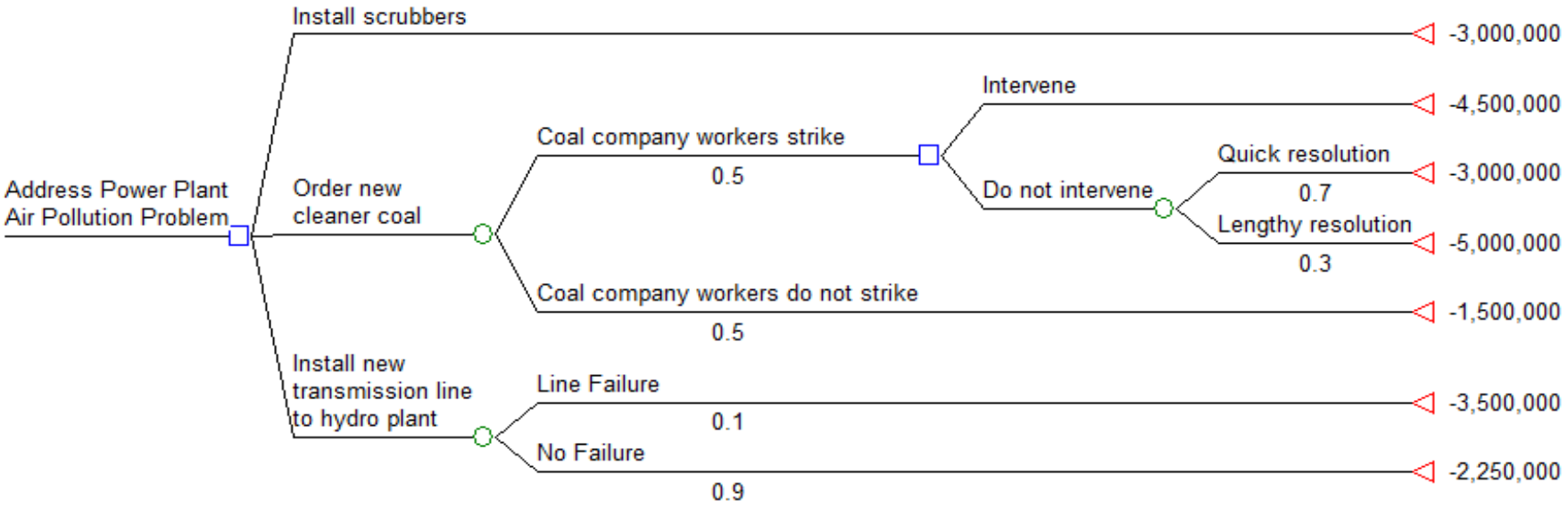


Going through the algorithm from right to left:

$$EU\ (\text{New coal strike with no intervention}) = 0.7(-3M)+0.3(-5M) = -3,600,000$$
$$EU\ (\text{Strike intervention?}) = max\ (-3.6M, -4.5M)) = -3,600,000$$

So the utility company should not intervene if a strike occurs.

Example: Power Plant Air Pollution Problem



$$EU\ (\text{Install scrubbers}) = -3,000,000$$
$$EU\ (\text{Ordering new coal}) = 0.5(-3.6M) + 0.5(-1.5M) = -2,550,000$$
$$EU\ (\text{Constructing new line}) = 0.1(-3.5M) + 0.9(-2.25M) = -2,375,000$$

Finally, $EU\ (\text{Installation Decision}) = max\ (-3M, -2.55M, -2.375M) = -2,375,000$

The result will be the utility company installing the new transmission line to the hydro plant because it has the lowest expected cost.